

## Problema 4

### Implementação

#### qr\_House.m

```
% Entradas:
% A - matriz de entrada (m x n)
% Saídas:
% U - matriz (m x m) contendo os vetores normais
% R - matriz (m x n) triangular superior
function [U, R] = qr_House(A)
    [m, n] = size(A);

    % Inicializar a matriz U com zeros
    U = zeros(m, m);

    for i = 1 : n
        % Extrair a coluna atual a partir da i-ésima linha até o final
        x = A(i:m, i);

        % Obtém o vetor normal ao hiperplano de reflexão
        if x(1) > 0
            x(1) = x(1) + norm(x, 2);
        else
            x(1) = x(1) - norm(x, 2);
        end

        u = x / norm(x, 2); % Normaliza o vetor
        U(i:m, i) = u; % Armazena o vetor em U

        % Aplica a transformação de Householder à submatriz de A
        A(i:m, i:n) = A(i:m, i:n) - 2*u*(u'*A(i:m, i:n));
    end

    R = triu(A); % Os valores abaixo da diagonal seriam próximos de 0
end
```

## qr\_House\_min.m

```
% Entradas:
%   A - matriz (m x n)
% Saídas:
%   U - matriz (m x n) contendo os vetores normais
%   R - matriz (m x k) triangular superior
function [U,R] = qr_House_min(A)
    [m, n] = size(A);

    % Determina a dimensão correta
    if m == n
        k = m - 1;
    else
        k = min(m,n); % Essa alteração abrange o caso onde m < n
    end

    % Inicializa matrizes
    R = A;
    U = zeros(m, k);

    for i = 1 : min(m,n)
        x = A(i:m, i);

        % Obtém o vetor normal ao hiperplano de reflexão
        if x(1) > 0
            x(1) = x(1) + norm(x, 2);
        else
            x(1) = x(1) - norm(x, 2);
        end

        u = x / norm(x,2); % Normaliza o vetor
        U(i:m, i) = u;     % Armazena o vetor em U

        % Aplica a transformação de Householder à submatriz de A
        A(i:m, i:n) = A(i:m, i:n) - 2*u*(u'*A(i:m, i:n));
    end

    R = triu(A(1:k, 1:n)); % Para que coincida com
end %endfunction
```

## constroi\_Q.m

```
% Entradas:
%   U - matriz (m x n) com vetores de Householder
% Saídas:
%   Q = matriz (m x n) ortogonal
function [Q] = constroi_Q(U)
    % Obtém dimensões de U e inicializa Q
    [m,n] = size(U);
    Q = eye(m,n);

    for i = 1 : n
        u = U(:,i);

        % Aplica a transformação de Householder pela direita Q*(H - u*u')
        Q = Q - 2*Q*(u*u');
    end
end
```

## Testes